Introduction to Machine Learning

Measures of Accuracy

Andres Mendez-Vazquez

February 25, 2019

Outline

Outline

Introduction

What did we see until now?

The design of learning machines from two main points:

- Statistical Point of View
- Linear Algebra and Optimization Point of View

Introduction

What did we see until now?

The design of learning machines from two main points:

- Statistical Point of View
- Linear Algebra and Optimization Point of View

Going back to the probability models

We might think in the machine to be learned as a function $g(x|\mathcal{D})$....

- Something as curve fitting...

Introduction

What did we see until now?

The design of learning machines from two main points:

- Statistical Point of View
- Linear Algebra and Optimization Point of View

Going back to the probability models

We might think in the machine to be learned as a function $g(x|\mathcal{D})$....

- Something as curve fitting...

Under a data set

$$\mathcal{D} = \{(x_i, y_i) | i = 1, 2, ..., N\} \quad (1)$$

Introduction

What did we see until now?

The design of learning machines from two main points:

- Statistical Point of View
- Linear Algebra and Optimization Point of View

Going back to the probability models

We might think in the machine to be learned as a function $g(\boldsymbol{x}|\mathcal{D})$....

- Something as curve fitting...

Under a data set

$$\mathcal{D} = \{(x_i, y_i) | i = 1, 2, ..., N\} \quad (1)$$

Introduction

What did we see until now?

The design of learning machines from two main points:

- Statistical Point of View
- Linear Algebra and Optimization Point of View

Going back to the probability models

We might think in the machine to be learned as a function $g(\boldsymbol{x}|\mathcal{D})$....

- Something as curve fitting...

Under a data set

$$\mathcal{D} = \{(x_i, y_i) \,|\, i = 1, 2, ..., N\} \tag{1}$$

Introduction

What did we see until now?

The design of learning machines from two main points:

- Statistical Point of View
- Linear Algebra and Optimization Point of View

Going back to the probability models

We might think in the machine to be learned as a function $g(\boldsymbol{x}|\mathcal{D})$....

- Something as curve fitting...

Under a data set

$$\mathcal{D} = \{(\boldsymbol{x}_i, y_i) \,|\, i = 1, 2, ..., N\} \tag{1}$$

Thus, we have that

Two main functions

- A function $g(\boldsymbol{x}|\mathcal{D})$ obtained using some algorithm!!!
- $E[y|\boldsymbol{x}]$ the optimal regression...

Thus, we have that

Two main functions

- A function $g(\boldsymbol{x}|\mathcal{D})$ obtained using some algorithm!!!
- $E[y|\boldsymbol{x}]$ the optimal regression...

Important

The key factor here is the dependence of the approximation on $\mathcal{D}$.

- This is the reason of studying fusion of information at decision level...

Thus, we have that

Two main functions

- A function $g(\boldsymbol{x}|\mathcal{D})$ obtained using some algorithm!!!
- $E[y|\boldsymbol{x}]$ the optimal regression...

Important

The key factor here is the dependence of the approximation on $\mathcal{D}$.

Why?

The approximation may be very good for a specific training data set but very bad for another.

- This is the reason of studying fusion of information at decision level...

Thus, we have that

Two main functions

- A function $g(\boldsymbol{x}|\mathcal{D})$ obtained using some algorithm!!!
- $E[y|\boldsymbol{x}]$ the optimal regression...

Important

The key factor here is the dependence of the approximation on $\mathcal{D}$.

Why?

The approximation may be very good for a specific training data set but very bad for another.

- This is the reason of studying fusion of information at decision level...

Thus, we have that

Two main functions

- A function $g(\boldsymbol{x}|\mathcal{D})$ obtained using some algorithm!!!
- $E[y|\boldsymbol{x}]$ the optimal regression...

Important

The key factor here is the dependence of the approximation on $\mathcal{D}$.

Why?

The approximation may be very good for a specific training data set but very bad for another.

- This is the reason of studying fusion of information at decision level...

Outline

We have the Bias-Variance

Our Final Equation

$$E_D\left(\left(g\left(\boldsymbol{x}|\mathcal{D}\right)-E\left[y|\boldsymbol{x}\right]\right)^2\right)=\underbrace{E_D\left(\left(g\left(\boldsymbol{x}|\mathcal{D}\right)-E_D\left[g\left(\boldsymbol{x}|\mathcal{D}\right)\right]\right)^2\right)}_{VARIANCE}+\underbrace{\left(E_D\left[g\left(\boldsymbol{x}|\mathcal{D}\right)\right]-E\left[y|\boldsymbol{x}\right]\right)^2}_{BIAS}$$

We have the Bias-Variance

Our Final Equation

$$E_D\left(\left(g\left(\boldsymbol{x}|\mathcal{D}\right)-E\left[y|\boldsymbol{x}\right]\right)^2\right)=\underbrace{E_D\left(\left(g\left(\boldsymbol{x}|\mathcal{D}\right)-E_D\left[g\left(\boldsymbol{x}|\mathcal{D}\right)\right]\right)^2\right)}_{VARIANCE}+\underbrace{\left(E_D\left[g\left(\boldsymbol{x}|\mathcal{D}\right)\right]-E\left[y|\boldsymbol{x}\right]\right)^2}_{BIAS}$$

Where the variance

It represents the measure of the error between our machine $g\left(\boldsymbol{x}|\mathcal{D}\right)$ and the expected output of the machine under $\boldsymbol{x}_i\sim p\left(\boldsymbol{x}|\Theta\right)$.

We have the Bias-Variance

Our Final Equation

$$E_D\left(\left(g\left(\boldsymbol{x}|\mathcal{D}\right)-E\left[y|\boldsymbol{x}\right]\right)^2\right)=\underbrace{E_D\left(\left(g\left(\boldsymbol{x}|\mathcal{D}\right)-E_D\left[g\left(\boldsymbol{x}|\mathcal{D}\right)\right]\right)^2\right)}_{VARIANCE}+\underbrace{\left(E_D\left[g\left(\boldsymbol{x}|\mathcal{D}\right)\right]-E\left[y|\boldsymbol{x}\right]\right)^2}_{BIAS}$$

Where the variance

It represents the measure of the error between our machine $g\left(\boldsymbol{x}|\mathcal{D}\right)$ and the expected output of the machine under $\boldsymbol{x}_i \sim p\left(\boldsymbol{x}|\Theta\right)$.

Where the bias

It represents the quadratic error between the expected output of the machine under $\boldsymbol{x}_i \sim p\left(\boldsymbol{x}|\Theta\right)$ and the expected output of the optimal regression.

We have the Bias-Variance

Our Final Equation

$$E_D\left(\left(g\left(\boldsymbol{x}|\mathcal{D}\right)-E\left[y|\boldsymbol{x}\right]\right)^2\right)=\underbrace{E_D\left(\left(g\left(\boldsymbol{x}|\mathcal{D}\right)-E_D\left[g\left(\boldsymbol{x}|\mathcal{D}\right)\right]\right)^2\right)}_{VARIANCE}+\underbrace{\left(E_D\left[g\left(\boldsymbol{x}|\mathcal{D}\right)\right]-E\left[y|\boldsymbol{x}\right]\right)^2}_{BIAS}$$

Where the variance

It represents the measure of the error between our machine $g\left(\boldsymbol{x}|\mathcal{D}\right)$ and the expected output of the machine under $\boldsymbol{x}_i \sim p\left(\boldsymbol{x}|\Theta\right)$.

Where the bias

It represents the quadratic error between the expected output of the machine under $\boldsymbol{x}_i \sim p\left(\boldsymbol{x}|\Theta\right)$ and the expected output of the optimal regression.

Remarks

We have then

Even if the estimator is unbiased, it can still result in a large mean square error due to a large variance term.

Remarks

We have then

Even if the estimator is unbiased, it can still result in a large mean square error due to a large variance term.

The situation is more dire in a finite set of data $\mathcal{D}$

We have then a trade-off:

1. Increasing the bias decreases the variance and vice versa.
2. This is known as the **bias–variance dilemma**.

Remarks

We have then

Even if the estimator is unbiased, it can still result in a large mean square error due to a large variance term.

The situation is more dire in a finite set of data $\mathcal{D}$

We have then a trade-off:

1. Increasing the bias decreases the variance and vice versa.
2. This is known as the **bias–variance dilemma**.

Remarks

We have then

Even if the estimator is unbiased, it can still result in a large mean square error due to a large variance term.

The situation is more dire in a finite set of data $\mathcal{D}$

We have then a trade-off:

1. Increasing the bias decreases the variance and vice versa.
2. This is known as the **bias–variance dilemma**.

Similar to...

Curve Fitting

If, for example, the adopted model is complex (many parameters involved) with respect to the number N, the model will fit the idiosyncrasies of the specific data set.

Similar to...

Curve Fitting

If, for example, the adopted model is complex (many parameters involved) with respect to the number N, the model will fit the idiosyncrasies of the specific data set.

Thus

Thus, it will result in low bias but will yield high variance, as we change from one data set to another data set.

- However, N is always finite!!!

Similar to...

Curve Fitting

If, for example, the adopted model is complex (many parameters involved) with respect to the number N, the model will fit the idiosyncrasies of the specific data set.

Thus

Thus, it will result in low bias but will yield high variance, as we change from one data set to another data set.

Furthermore

If N grows we can have a more complex model to be fitted which reduces bias and ensures low variance.

- However, N is always finite!!!

Similar to...

Curve Fitting

If, for example, the adopted model is complex (many parameters involved) with respect to the number N, the model will fit the idiosyncrasies of the specific data set.

Thus

Thus, it will result in low bias but will yield high variance, as we change from one data set to another data set.

Furthermore

If N grows we can have a more complex model to be fitted which reduces bias and ensures low variance.

- However, N is always finite!!!

Thus

You always need to compromise

However, you always have some a priori knowledge about the data

Allowing you to impose restrictions

Lowering the bias and the variance

Nevertheless

We have the following example to grasp better the bothersome **bias–variance dilemma**.

Thus

You always need to compromise

However, you always have some a priori knowledge about the data

Allowing you to impose restrictions

Lowering the bias and the variance

Nevertheless

We have the following example to grasp better the bothersome **bias–variance dilemma**.

Thus

You always need to compromise

However, you always have some a priori knowledge about the data

Allowing you to impose restrictions

Lowering the bias and the variance

Nevertheless

We have the following example to grasp better the bothersome **bias–variance dilemma**.

For this

Assume

The data is generated by the following function

$$y = f(x) + \epsilon,$$
$$\epsilon \sim \mathcal{N}\left(0, \sigma_\epsilon^2\right)$$

We know that

The optimum regressor is $E[y|x] = f(x)$

Furthermore

Assume that the randomness in the different training sets, $\mathcal{D}$, is due to the y_i's (Affected by noise), while the respective points, x_i, are fixed.

For this

Assume

The data is generated by the following function

$$y = f(x) + \epsilon,$$
$$\epsilon \sim \mathcal{N}\left(0, \sigma_\epsilon^2\right)$$

We know that

The optimum regressor is $E[y|x] = f(x)$

Furthermore

Assume that the randomness in the different training sets, $\mathcal{D}$, is due to the y_i's (Affected by noise), while the respective points, x_i, are fixed.

For this

Assume

The data is generated by the following function

$$y = f(x) + \epsilon,$$
$$\epsilon \sim \mathcal{N}\left(0, \sigma_\epsilon^2\right)$$

We know that

The optimum regressor is $E[y|x] = f(x)$

Furthermore

Assume that the randomness in the different training sets, $\mathcal{D}$, is due to the y_i's (Affected by noise), while the respective points, x_i, are fixed.

Outline

Sampling the Space

Imagine that $\mathcal{D} \subset [x_1, x_2]$ in which x lies

For example, you can choose $x_i = x_1 + \frac{x_2 - x_1}{N-1}(i-1)$ with $i = 1, 2, ..., N$

Case 1

Choose the estimate of $f(x)$, $g(x|\mathcal{D})$, to be independent of $\mathcal{D}$

For example, $g(x) = w_1 x + w_0$

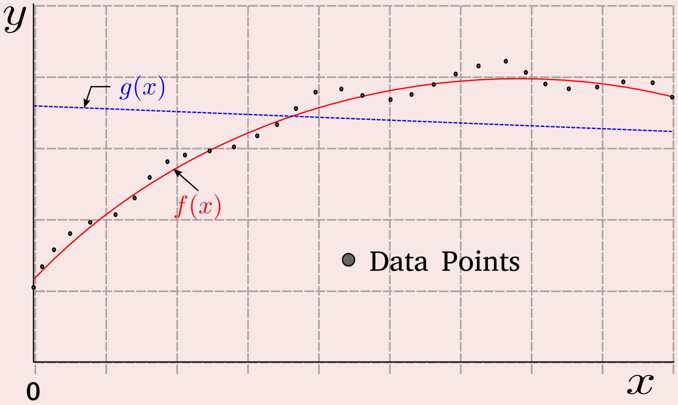
For example, the points are spread around $(x, f(x))$

Case 1

Choose the estimate of $f(x)$, $g(x|\mathcal{D})$, to be independent of $\mathcal{D}$

For example, $g(x) = w_1 x + w_0$

For example, the points are spread around $(x, f(x))$



Case 2

In the other hand

Now, $g_1(x)$ corresponds to a polynomial of high degree so it can pass through each training point in $\mathcal{D}$.

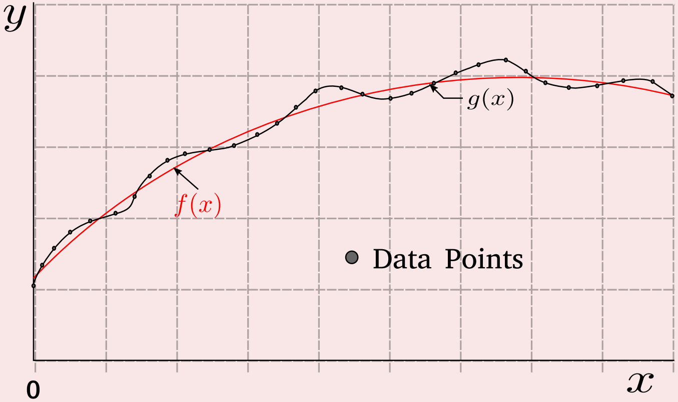
Example of $g_1(x)$

Case 2

In the other hand

Now, $g_1(x)$ corresponds to a polynomial of high degree so it can pass through each training point in $\mathcal{D}$.

Example of $g_1(x)$

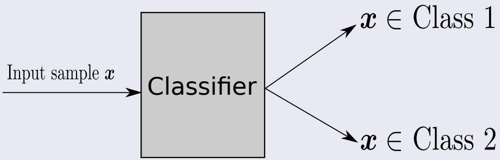


Outline

Sooner of Latter you need to know how efficient is your algorithm

Thus, we need a measures of accuracy

Thus, we begin with the classic classifier for two classes



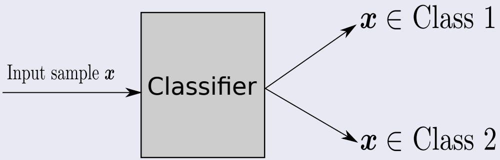
Here

A dataset used for performance evaluation is called a **test dataset**.

Sooner of Latter you need to know how efficient is your algorithm

Thus, we need a measures of accuracy

Thus, we begin with the classic classifier for two classes



Here

A dataset used for performance evaluation is called a **test dataset.**

Therefore

It is a good idea to build a measure of performance

For this, we can use the idea of error in statistics.

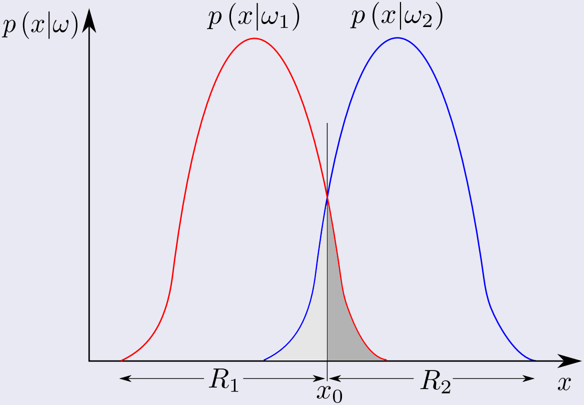
Do you remember?

Therefore

It is a good idea to build a measure of performance

For this, we can use the idea of error in statistics.

Do you remember?



Outline

α error

Definition (Type I Error - False Positive)

α is the probability that the test will lead to the rejection of the hypothesis H_0 when that hypothesis is true.

- You have a device that fails $\alpha = 0.05$ meaning that it fails 5% of the time
- This says that you ha low chance of a wrong circuit

α error

Definition (Type I Error - False Positive)

α is the probability that the test will lead to the rejection of the hypothesis H_0 when that hypothesis is true.

Example

1. H_0 : "You have a device that produce circuits with no error"
2. You have a device that fails $\alpha = 0.05$ meaning that it fails 5 of the time.
3. This says that you ha low chance of a wrong circuit.

α error

Definition (Type I Error - False Positive)

α is the probability that the test will lead to the rejection of the hypothesis H_0 when that hypothesis is true.

Example

1. H_0 : "You have a device that produce circuits with no error"
2. You have a device that fails $\alpha = 0.05$ meaning that it fails 5 of the time.
3. This says that you ha low chance of a wrong circuit.

α error

Definition (Type I Error - False Positive)

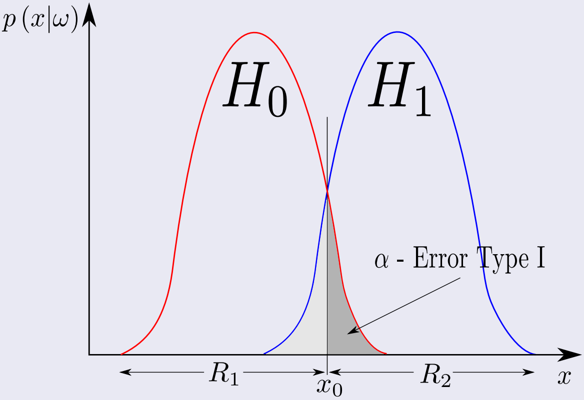
α is the probability that the test will lead to the rejection of the hypothesis H_0 when that hypothesis is true.

Example

1. H_0 : "You have a device that produce circuits with no error"
2. You have a device that fails $\alpha = 0.05$ meaning that it fails 5 of the time.
3. This says that you ha low chance of a wrong circuit.

Basically

We have



β error

Definition (Type II Error - False Negative)

β is the probability that the test will lead to the rejection of the hypothesis H_1 when that hypothesis is true.

β error

Definition (Type II Error - False Negative)

β is the probability that the test will lead to the rejection of the hypothesis H_1 when that hypothesis is true.

Example

1. H_1: "Adding fluoride to toothpaste protects against cavities."
2. Then $\beta = 0.05$ meaning that you have a chance of 5 of the time.
3. This says that you ha low chance of having a cavity using fluoride in the water.

β error

Definition (Type II Error - False Negative)

β is the probability that the test will lead to the rejection of the hypothesis H_1 when that hypothesis is true.

Example

1. H_1: “Adding fluoride to toothpaste protects against cavities."
2. Then $\beta = 0.05$ meaning that you have a chance of 5 of the time.
3. This says that you ha low chance of having a cavity using fluoride in the water.

β error

Definition (Type II Error - False Negative)

β is the probability that the test will lead to the rejection of the hypothesis H_1 when that hypothesis is true.

Example

1. H_1: "Adding fluoride to toothpaste protects against cavities."
2. Then $\beta = 0.05$ meaning that you have a chance of 5 of the time.
3. This says that you ha low chance of having a cavity using fluoride in the water.

Outline

This can be seen as a table

Confusion Matrix from the statistic point of view

Table of error types		Null Hypothesis H_0	
		True	False
Decision about H_0	Reject	Type I Error - α **False Positive**	Correct Inference **True Positive**
	Fail to reject	Correct Inference **True Negative**	Type II Error - β **False Negative**

In the case of two classes, we have

Confusion Matrix from the Machine Learning point of view

		Actual Class	
		Positive	Negative
Predicted Classes	Positive	**True Positive (TP)**	**False Positives (FP)**
	Negative	**False Negatives (FN)**	**True Negatives (TN)**

Outline

Accuracy

Definition

The proportion of getting correct classification of the Positive and Negative classes.

Thus

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + FN + TN}$$

Problem - accuracy assumes equal cost for both kinds of errors

Is 99% accuracy good, bad or terrible? It depends on the problem.

Accuracy

Definition

The proportion of getting correct classification of the Positive and Negative classes.

Thus

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + FN + TN}$$

Problem - accuracy assumes equal cost for both kinds of errors

Is 99% accuracy good, bad or terrible? It depends on the problem.

Accuracy

Definition

The proportion of getting correct classification of the Positive and Negative classes.

Thus

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + FN + TN}$$

Problem - accuracy assumes equal cost for both kinds of errors

Is 99% accuracy good, bad or terrible? It depends on the problem.

True Positive Rate

Also called

Sensitivity or Recall Rate

Defined as

True Positive Rate is the proportion of getting a correct classification of the Positive Class vs the True Positive and False Negatives.

$$\text{True Positive Rate} = \frac{TP}{TP + FN}$$

True Positive Rate

Also called

Sensitivity or Recall Rate

Defined as

True Positive Rate is the proportion of getting a correct classification of the Positive Class vs the True Positive and False Negatives.

$$\text{True Positive Rate} = \frac{TP}{TP + FN}$$

True Negative Rate

Also known as

Specificity

Defined as

It is the proportion of True Negative vs the elements classified as True negatives.

$$\text{True Negative Rate} = \frac{TN}{FP + TN}$$

True Negative Rate

Also known as

Specificity

Defined as

It is the proportion of True Negative vs the elements classified as True negatives.

$$\text{True Negative Rate} = \frac{TN}{FP + TN}$$

Precision

Also known as

Positive Predictive Value

Defined as

The proportion of the elements classified as true positive vs the total of all the real true positives.

$$\text{Precision Predicted Value} = \frac{TP}{FP + TP}$$

Precision

Also known as

Positive Predictive Value

Defined as

The proportion of the elements classified as true positive vs the total of all the real true positives.

$$\text{Precision Predicted Value} = \frac{TP}{FP + TP}$$

Significance Level

Also known as

False Positive Rate.

Defined as

False Positive Rate is the probability of getting an incorrect classification of the Positive Class vs the True Negative and the False Positive.

$$\text{False positive rate} = \frac{FP}{TN + FP}$$

Significance Level

Also known as

False Positive Rate.

Defined as

False Positive Rate is the probability of getting an incorrect classification of the Positive Class vs the True Negative and the False Positive.

$$\text{False positive rate} = \frac{FP}{TN + FP}$$

Outline

We can do better than these simple measures of accuracy

Given these initial measures of validity

it is possible to obtain a more precise model evaluation, the ROC curves.

The ROC Curves plot

It is a model-wide evaluation measure that is based on two basic evaluation measures:

1. **Specificity** is a performance measure of the whole negative part of a dataset.
2. **Sensitivity** is a performance measure of the whole positive part.

We can do better than these simple measures of accuracy

Given these initial measures of validity

it is possible to obtain a more precise model evaluation, the ROC curves.

The ROC Curves plot

It is a model-wide evaluation measure that is based on two basic evaluation measures:

1. **Specificity** is a performance measure of the whole negative part of a dataset.
2. **Sensitivity** is a performance measure of the whole positive part.

What the ROC Curves uses

We have a plot where

The ROC plot uses specificity on the x-axis and sensitivity on the y-axis.

1. Two adjacent ROC points can be connected by a straight line
2. The curve starts at (0.0, 0.0) and ends at (1.0, 1.0)

What the ROC Curves uses

We have a plot where

The ROC plot uses specificity on the x-axis and sensitivity on the y-axis.

Basically

False Positive Rate (FPR) is identical with specificity, and True Positive Rate (TPR) is identical with sensitivity.

1. Two adjacent ROC points can be connected by a straight line
2. The curve starts at (0.0, 0.0) and ends at (1.0, 1.0)

What the ROC Curves uses

We have a plot where

The ROC plot uses specificity on the x-axis and sensitivity on the y-axis.

Basically

False Positive Rate (FPR) is identical with specificity, and True Positive Rate (TPR) is identical with sensitivity.

Then

1. A ROC curve is created by connecting all ROC points of a classier in the ROC space.
2. Two adjacent ROC points can be connected by a straight line.
3. The curve starts at (0.0, 0.0) and ends at (1.0, 1.0).

What the ROC Curves uses

We have a plot where

The ROC plot uses specificity on the x-axis and sensitivity on the y-axis.

Basically

False Positive Rate (FPR) is identical with specificity, and True Positive Rate (TPR) is identical with sensitivity.

Then

1. A ROC curve is created by connecting all ROC points of a classier in the ROC space.
2. Two adjacent ROC points can be connected by a straight line.
3. The curve starts at (0.0, 0.0) and ends at (1.0, 1.0).

What the ROC Curves uses

We have a plot where

The ROC plot uses specificity on the x-axis and sensitivity on the y-axis.

Basically

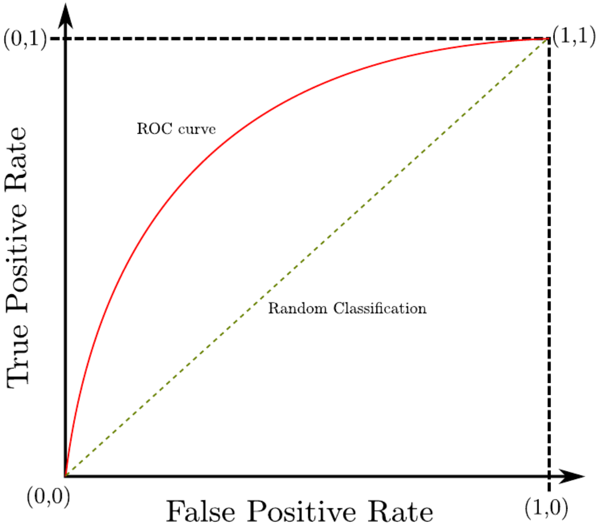
False Positive Rate (FPR) is identical with specificity, and True Positive Rate (TPR) is identical with sensitivity.

Then

1. A ROC curve is created by connecting all ROC points of a classier in the ROC space.
2. Two adjacent ROC points can be connected by a straight line.
3. The curve starts at (0.0, 0.0) and ends at (1.0, 1.0).

Outline

For Example



(0,1)
(1,1)
ROC curve
Random Classification
True Positive Rate
(0,0)
False Positive Rate
(1,0)

Outline

We have

Algorithm ROC point generation

Input: L, the set of test examples; $f(i)$, the probabilistic classifier estimate that example i is positive; P and N, the number of positive and negative examples.

Ouput: R, a list of ROC points increasing by false positive rate.

1. $L_{sorted} \leftarrow L$ sorted decreasing by f scores
2. $FP \leftarrow TP \leftarrow 0; R \leftarrow \langle\rangle; fprev \leftarrow -\infty; i \leftarrow 1$
3. **while** $i \leq |L_{sorted}|$
4.   **if** $f(i) \neq f_{prev}$ **then**
5. $R.append\left(\frac{FP}{N}, \frac{TP}{P}\right)$
6. $fprev \leftarrow f(i)$
7.   **if** L_{sorted} is a positive example **then** $TP = TP + 1$
8.   **else** $FP = FP + 1$
9. $i \leftarrow i + 1$
10. $R.append\left(\frac{FP}{N}, \frac{TP}{P}\right)$, this is $(1, 1)$

We have

Algorithm ROC point generation

Input: L, the set of test examples; $f(i)$, the probabilistic classifier estimate that example i is positive; P and N, the number of positive and negative examples.

Ouput: R, a list of ROC points increasing by false positive rate.

```
L_sorted ← L sorted decreasing by f scores
FP ← TP ← 0; R ← ⟨⟩; fprev ← −∞; i ← 1
while i ≤ |L_sorted|
    if f(i) ≠ f_prev then
        R.append(FP/N, TP/P)
        fprev ← f(i)
    if L_sorted is a positive example then TP = TP + 1
    else FP = FP + 1
    i ← i + 1
R.append(FP/N, TP/P), this is (1, 1)
```

We have

Algorithm ROC point generation

Input: L, the set of test examples; $f(i)$, the probabilistic classifier estimate that example i is positive; P and N, the number of positive and negative examples.

Ouput: R, a list of ROC points increasing by false positive rate.

1. $L_{sorted} \leftarrow L$ **sorted decreasing by f scores**
2. $FP \leftarrow TP \leftarrow 0; R \leftarrow \langle\rangle; fprev \leftarrow -\infty; i \leftarrow 1$
3. **while** $i \leq |L_{sorted}|$
4.   **if** $f(i) \neq f_{prev}$ **then**
5. $R.append\left(\frac{FP}{N}, \frac{TP}{P}\right)$
6. $fprev \leftarrow f(i)$
7.   **if** L_{sorted} **is a positive example then** $TP = TP + 1$
8.   **else** $FP = FP + 1$
9. $i \leftarrow i + 1$
10. $R.append\left(\frac{FP}{N}, \frac{TP}{P}\right)$, **this is** $(1, 1)$

We have

Algorithm ROC point generation

Input: L, the set of test examples; $f(i)$, the probabilistic classifier estimate that example i is positive; P and N, the number of positive and negative examples.

Ouput: R, a list of ROC points increasing by false positive rate.

1. $L_{sorted} \leftarrow L$ **sorted decreasing by** f **scores**
2. $FP \leftarrow TP \leftarrow 0; R \leftarrow \langle\rangle; fprev \leftarrow -\infty; i \leftarrow 1$
3. while $i \leq |L_{sorted}|$
4. if $f(i) \neq f_{prev}$ then
5. $R.append\left(\frac{FP}{N}, \frac{TP}{P}\right)$
6. $fprev \leftarrow f(i)$
7. if L_{sorted} is a positive example then $TP = TP + 1$
8. else $FP = FP + 1$
9. $i \leftarrow i + 1$
10. $R.append\left(\frac{FP}{N}, \frac{TP}{P}\right)$, this is $(1, 1)$

We have

Algorithm ROC point generation

Input: L, the set of test examples; $f(i)$, the probabilistic classifier estimate that example i is positive; P and N, the number of positive and negative examples.

Ouput: R, a list of ROC points increasing by false positive rate.

1. $L_{sorted} \leftarrow L$ **sorted decreasing by** f **scores**
2. $FP \leftarrow TP \leftarrow 0; R \leftarrow \langle\rangle; fprev \leftarrow -\infty; i \leftarrow 1$
3. **while** $i \leq |L_{sorted}|$
4. if $f(i) \neq f_{prev}$ then
5. $R.append\left(\frac{FP}{N}, \frac{TP}{P}\right)$
6. $fprev \leftarrow f(i)$
7. if L_{sorted} is a positive example then $TP = TP + 1$
8. else $FP = FP + 1$
9. $i \leftarrow i + 1$
10. $R.append\left(\frac{FP}{N}, \frac{TP}{P}\right)$, this is $(1, 1)$

We have

Algorithm ROC point generation

Input: L, the set of test examples; $f(i)$, the probabilistic classifier estimate that example i is positive; P and N, the number of positive and negative examples.

Ouput: R, a list of ROC points increasing by false positive rate.

1. $L_{sorted} \leftarrow L$ **sorted decreasing by** f **scores**
2. $FP \leftarrow TP \leftarrow 0; R \leftarrow \langle\rangle; fprev \leftarrow -\infty; i \leftarrow 1$
3. **while** $i \leq |L_{sorted}|$
4.   **if** $f(i) \neq f_{prev}$ **then**
5. $R.append\left(\frac{FP}{N}, \frac{TP}{P}\right)$
6. $fprev \leftarrow f(i)$
7.   **if** L_{sorted} **is a positive example then** $TP = TP + 1$
8.   **else** $FP = FP + 1$
9. $i \leftarrow i + 1$
10. $R.append\left(\frac{FP}{N}, \frac{TP}{P}\right)$, **this is** $(1, 1)$

We have

Algorithm ROC point generation

Input: L, the set of test examples; $f(i)$, the probabilistic classifier estimate that example i is positive; P and N, the number of positive and negative examples.

Ouput: R, a list of ROC points increasing by false positive rate.

$L_{sorted} \leftarrow L$ **sorted decreasing by** f **scores**

$FP \leftarrow TP \leftarrow 0; R \leftarrow \langle\rangle; fprev \leftarrow -\infty; i \leftarrow 1$

while $i \leq |L_{sorted}|$

  if $f(i) \neq f_{prev}$ **then**

    $R.append\left(\frac{FP}{N}, \frac{TP}{P}\right)$

    $fprev \leftarrow f(i)$

  if L_{sorted} **is a positive example then** $TP = TP + 1$

  else $FP = FP + 1$

  $i \leftarrow i + 1$

$R.append\left(\frac{FP}{N}, \frac{TP}{P}\right)$, **this is** $(1, 1)$

We have

Algorithm ROC point generation

Input: L, the set of test examples; $f(i)$, the probabilistic classifier estimate that example i is positive; P and N, the number of positive and negative examples.

Ouput: R, a list of ROC points increasing by false positive rate.

1. $L_{sorted} \leftarrow L$ **sorted decreasing by** f **scores**
2. $FP \leftarrow TP \leftarrow 0; R \leftarrow \langle\rangle; fprev \leftarrow -\infty; i \leftarrow 1$
3. **while** $i \leq |L_{sorted}|$
4.   **if** $f(i) \neq f_{prev}$ **then**
5. $R.append\left(\frac{FP}{N}, \frac{TP}{P}\right)$
6. $fprev \leftarrow f(i)$
7.   **if** L_{sorted} **is a positive example then** $TP = TP + 1$
8.   **else** $FP = FP + 1$
9. $i \leftarrow i + 1$
10. $R.append\left(\frac{FP}{N}, \frac{TP}{P}\right)$, **this is** $(1, 1)$

We have

Algorithm ROC point generation

Input: L, the set of test examples; $f(i)$, the probabilistic classifier estimate that example i is positive; P and N, the number of positive and negative examples.

Ouput: R, a list of ROC points increasing by false positive rate.

$L_{sorted} \leftarrow L$ **sorted decreasing by** f **scores**
$FP \leftarrow TP \leftarrow 0; R \leftarrow \langle\rangle; fprev \leftarrow -\infty; i \leftarrow 1$
while $i \leq |L_{sorted}|$
    if $f(i) \neq f_{prev}$ **then**
        $R.append\left(\frac{FP}{N}, \frac{TP}{P}\right)$
        $fprev \leftarrow f(i)$
    if L_{sorted} **is a positive example then** $TP = TP + 1$
    else $FP = FP + 1$
    $i \leftarrow i + 1$
$R.append\left(\frac{FP}{N}, \frac{TP}{P}\right)$, **this is** $(1, 1)$

We have

Algorithm ROC point generation

Input: L, the set of test examples; $f(i)$, the probabilistic classifier estimate that example i is positive; P and N, the number of positive and negative examples.

Ouput: R, a list of ROC points increasing by false positive rate.

$L_{sorted} \leftarrow L$ **sorted decreasing by** f **scores**

$FP \leftarrow TP \leftarrow 0; R \leftarrow \langle\rangle ; fprev \leftarrow -\infty; i \leftarrow 1$

while $i \leq |L_{sorted}|$

    if $f(i) \neq f_{prev}$ **then**

        $R.append\left(\frac{FP}{N}, \frac{TP}{P}\right)$

        $fprev \leftarrow f(i)$

    if L_{sorted} **is a positive example then** $TP = TP + 1$

    else $FP = FP + 1$

    $i \leftarrow i + 1$

$R.append\left(\frac{FP}{N}, \frac{TP}{P}\right)$, **this is** $(1,1)$

We have

Algorithm ROC point generation

Input: L, the set of test examples; $f(i)$, the probabilistic classifier estimate that example i is positive; P and N, the number of positive and negative examples.

Ouput: R, a list of ROC points increasing by false positive rate.

$L_{sorted} \leftarrow L$ **sorted decreasing by** f **scores**
$FP \leftarrow TP \leftarrow 0; R \leftarrow \langle\rangle; fprev \leftarrow -\infty; i \leftarrow 1$
while $i \leq |L_{sorted}|$
    if $f(i) \neq f_{prev}$ **then**
        $R.append\left(\frac{FP}{N}, \frac{TP}{P}\right)$
        $fprev \leftarrow f(i)$
    if L_{sorted} **is a positive example then** $TP = TP + 1$
    else $FP = FP + 1$
    $i \leftarrow i + 1$
$R.append\left(\frac{FP}{N}, \frac{TP}{P}\right)$, **this is** $(1, 1)$

We have

Algorithm ROC point generation

Input: L, the set of test examples; $f(i)$, the probabilistic classifier estimate that example i is positive; P and N, the number of positive and negative examples.

Ouput: R, a list of ROC points increasing by false positive rate.

1. $L_{sorted} \leftarrow L$ **sorted decreasing by** f **scores**
2. $FP \leftarrow TP \leftarrow 0; R \leftarrow \langle \rangle ; fprev \leftarrow -\infty; i \leftarrow 1$
3. **while** $i \leq |L_{sorted}|$
4.   **if** $f(i) \neq f_{prev}$ **then**
5. $R.append\left(\frac{FP}{N}, \frac{TP}{P}\right)$
6. $fprev \leftarrow f(i)$
7.   **if** L_{sorted} **is a positive example then** $TP = TP + 1$
8.   **else** $FP = FP + 1$
9. $i \leftarrow i + 1$
10. $R.append\left(\frac{FP}{N}, \frac{TP}{P}\right)$, **this is** $(1, 1)$

Thus

Thus

Thus, after generating the ROC Curve it is possible to use several metrics to validate using the ROC curves.

A Partial List is

1. Area Under the Curve (AUC)
2. Equal Error Rate (EER)
3. Likelihood Ratio

Thus

Thus

Thus, after generating the ROC Curve it is possible to use several metrics to validate using the ROC curves.

A Partial List is

1. Area Under the Curve (AUC)
2. Equal Error Rate (EER)
3. Likelihood Ratio

Outline

A Simple Definition

We have

$$AUC = \int ROC\left(p\right)dp = \sum_{i=1}^{N} ROC\left(f\left(\frac{1}{i}\right)\right)\left[\frac{i}{N} - \frac{i-1}{N}\right]$$

This equation has the following meaning

- The probability that a randomly selected observation X from the **positive class** would have a higher score than a randomly selected observation Y from the **negative class.**

$$P\left(X > Y\right)$$

Thus

The AUC gives the mean **true positive** rate averaged uniformly across the **false positive** rate.

A Simple Definition

We have

$$AUC = \int ROC\left(p\right) dp = \sum_{i=1}^{N} ROC\left(f\left(\frac{1}{i}\right)\right)\left[\frac{i}{N} - \frac{i-1}{N}\right]$$

This equation has the following meaning

- The probability that a randomly selected observation X from the **positive class** would have a higher score than a randomly selected observation Y from the **negative class.**

$$P\left(X > Y\right)$$

Thus

The AUC gives the mean **true positive** rate averaged uniformly across the **false positive** rate.

A Simple Definition

We have

$$AUC = \int ROC\,(p)\,dp = \sum_{i=1}^{N} ROC\left(f\left(\frac{1}{i}\right)\right)\left[\frac{i}{N} - \frac{i-1}{N}\right]$$

This equation has the following meaning

- The probability that a randomly selected observation X from the **positive class** would have a higher score than a randomly selected observation Y from the **negative class.**

$$P\,(X > Y)$$

Thus

The AUC gives the mean **true positive** rate averaged uniformly across the **false positive** rate.

Outline

What we want

We want to measure

- A quality measure to measure different classifiers (for different parameter values).

We call that as

$$R(f) = E_{\mathcal{D}}\left[L\left(y, f\left(x\right)\right)\right]. \quad (2)$$

Example: $L\left(y, f\left(x\right)\right) = \left\|y - f\left(x\right)\right\|_2^2$

More precisely

- For different values γ_j of the parameter, we train a classifier $f\left(x|\gamma_j\right)$ on the training set.

What we want

We want to measure

- A quality measure to measure different classifiers (for different parameter values).

We call that as

$$R(f) = E_{\mathcal{D}}\left[L\left(y, f\left(\boldsymbol{x}\right)\right)\right]. \tag{2}$$

Example: $L\left(y, f\left(x\right)\right) = \left\|y - f\left(\boldsymbol{x}\right)\right\|_2^2$

More precisely

- For different values γ_j of the parameter, we train a classifier $f\left(x|\gamma_j\right)$ on the training set.

What we want

We want to measure

- A quality measure to measure different classifiers (for different parameter values).

We call that as

$$R(f) = E_{\mathcal{D}}\left[L\left(y, f\left(\boldsymbol{x}\right)\right)\right]. \tag{2}$$

Example: $L\left(y, f\left(x\right)\right) = \left\|y - f\left(\boldsymbol{x}\right)\right\|_2^2$

More precisely

- For different values γ_j of the parameter, we train a classifier $f\left(\boldsymbol{x}|\gamma_j\right)$ on the training set.

Then, calculate the empirical Risk

Empirical Risk

- We use the validation set to estimate

$$\hat{R}\left(f\left(x|\gamma_i\right)\right) = \frac{1}{N_v}\sum_{i=1}^{N_v} L\left(y_i f\left(\boldsymbol{x}_i|\gamma_j\right)\right) \tag{3}$$

- Re-train the classifier with the best parameter γ^* on all data except the test set (i.e. train + validation data).
- Report error estimate $\hat{R}\left(f\left(x|\gamma^*\right)\right)$ computed on the test set

Then, calculate the empirical Risk

Empirical Risk

- We use the validation set to estimate

$$\hat{R}\left(f\left(x|\gamma_i\right)\right) = \frac{1}{N_v}\sum_{i=1}^{N_v} L\left(y_i f\left(\boldsymbol{x}_i|\gamma_j\right)\right) \tag{3}$$

Thus, you follow the following procedure

1. Select the value γ^* which achieves the smallest estimated error.
2. Re-train the classifier with the best parameter γ^* on all data except the test set (i.e. train + validation data).
3. Report error estimate $\hat{R}\left(f\left(x|\gamma^*\right)\right)$ computed on the test set.

Then, calculate the empirical Risk

Empirical Risk

- We use the validation set to estimate

$$\hat{R}\left(f\left(x|\gamma_i\right)\right)=\frac{1}{N_v}\sum_{i=1}^{N_v}L\left(y_i f\left(\boldsymbol{x}_i|\gamma_j\right)\right) \tag{3}$$

Thus, you follow the following procedure

1. Select the value γ^* which achieves the smallest estimated error.
2. Re-train the classifier with the best parameter γ^* on all data except the test set (i.e. train + validation data).
3. Report error estimate $\hat{R}\left(f\left(x|\gamma^*\right)\right)$ computed on the test set.

Then, calculate the empirical Risk

Empirical Risk

- We use the validation set to estimate

$$\hat{R}\left(f\left(x|\gamma_i\right)\right) = \frac{1}{N_v}\sum_{i=1}^{N_v} L\left(y_i f\left(\boldsymbol{x}_i|\gamma_j\right)\right) \tag{3}$$

Thus, you follow the following procedure

1. Select the value γ^* which achieves the smallest estimated error.
2. Re-train the classifier with the best parameter γ^* on all data except the test set (i.e. train + validation data).
3. Report error estimate $\hat{R}\left(f\left(x|\gamma^*\right)\right)$ computed on the test set.

Outline

K-Cross Validation

Procedure

To estimate the risk of a classifier f:

1. Split data into K equally sized parts (called "folds").
2. Train an instance f_k of the classifier, using all folds except fold k as training data.
3. Compute the cross validation (CV) estimate:

$$\hat{R}_{CV}\left(f\left(x|\gamma_i\right)\right) = \frac{1}{N_v}\sum_{i=1}^{N_v} L\left(y_i f\left(x_{k(i)}|\gamma_j\right)\right) \tag{4}$$

where $k(i)$ is the fold containing x_i.

K-Cross Validation

Procedure

To estimate the risk of a classifier f:

1. Split data into K equally sized parts (called "folds").
2. Train an instance f_k of the classifier, using all folds except fold k as training data.
3. Compute the cross validation (CV) estimate:

$$\hat{R}_{CV}\left(f\left(x|\gamma_i\right)\right) = \frac{1}{N_v}\sum_{i=1}^{N_v} L\left(y_i f\left(x_{k(i)}|\gamma_j\right)\right) \quad (4)$$

where $k(i)$ is the fold containing x_i.

K-Cross Validation

Procedure

To estimate the risk of a classifier f:

1. Split data into K equally sized parts (called "folds").
2. Train an instance f_k of the classifier, using all folds except fold k as training data.
3. Compute the cross validation (CV) estimate:

$$\hat{R}_{CV}\left(f\left(x|\gamma_i\right)\right) = \frac{1}{N_v}\sum_{i=1}^{N_v} L\left(y_i f\left(x_{k(i)}|\gamma_j\right)\right) \tag{4}$$

where $k\left(i\right)$ is the fold containing x_i.

K-Cross Validation

Procedure

To estimate the risk of a classifier f:

1. Split data into K equally sized parts (called "folds").
2. Train an instance f_k of the classifier, using all folds except fold k as training data.
3. Compute the cross validation (CV) estimate:

$$\hat{R}_{CV}\left(f\left(x|\gamma_i\right)\right) = \frac{1}{N_v}\sum_{i=1}^{N_v} L\left(y_i f\left(\boldsymbol{x}_{k(i)}|\gamma_j\right)\right) \tag{4}$$

where $k\left(i\right)$ is the fold containing $\boldsymbol{x}_i$.

Example

$K = 5$

- Randomize the data and split the data in five folds

Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
1	2	3	4	5

- Then select one fold for testing

Therefore

- Cross validation procedure does not involve the test data.

SPLIT THIS PART

Train Data + Validation Data	Test

- The other folds not used for testing are split into Train Data and Validation data (80% - 20% or 90% - 10%)

Example

$K = 5$

- Randomize the data and split the data in five folds

Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
1	2	3	4	5

- Then select one fold for testing

Therefore

- Cross validation procedure does not involve the test data.

$\overbrace{\text{Train Data + Validation Data}}^{\text{SPLIT THIS PART}}$	Test

- The other folds not used for testing are split into Train Data and Validation data (80% - 20% or 90% - 10%)

Outline

How to choose K

Extremal cases

- $K = N$, called leave one out cross validation (loocv)
- $K = 2$

An often-cited problem with loocv is that we have to train many $(= N)$ classifiers, but there is also a deeper problem.

How to choose K

Extremal cases

- $K = N$, called leave one out cross validation (loocv)
- $K = 2$

An often-cited problem with loocv is that we have to train many ($= N$) classifiers, but there is also a deeper problem.

Argument 1: K should be small, e.g. $K = 2$

1. Unless we have a lot of data, variance between two distinct training sets may be considerable.
2. Important concept: By removing substantial parts of the sample in turn and at random, we can simulate this variance.
3. By removing a single point (loocv), we cannot make this variance visible.

How to choose K

Extremal cases

- $K = N$, called leave one out cross validation (loocv)
- $K = 2$

An often-cited problem with loocv is that we have to train many $(= N)$ classifiers, but there is also a deeper problem.

Argument 1: K should be small, e.g. $K = 2$

1. Unless we have a lot of data, variance between two distinct training sets may be considerable.
2. Important concept: By removing substantial parts of the sample in turn and at random, we can simulate this variance.
3. By removing a single point (loocv), we cannot make this variance visible.

How to choose K

Extremal cases

- $K = N$, called leave one out cross validation (loocv)
- $K = 2$

An often-cited problem with loocv is that we have to train many ($= N$) classifiers, but there is also a deeper problem.

Argument 1: K should be small, e.g. $K = 2$

1. Unless we have a lot of data, variance between two distinct training sets may be considerable.
2. Important concept: By removing substantial parts of the sample in turn and at random, we can simulate this variance.
3. By removing a single point (loocv), we cannot make this variance visible.

How to choose K

Extremal cases

- $K = N$, called leave one out cross validation (loocv)
- $K = 2$

An often-cited problem with loocv is that we have to train many $(= N)$ classifiers, but there is also a deeper problem.

Argument 1: K should be small, e.g. $K = 2$

1. Unless we have a lot of data, variance between two distinct training sets may be considerable.
2. Important concept: By removing substantial parts of the sample in turn and at random, we can simulate this variance.
3. By removing a single point (loocv), we cannot make this variance visible.

How to choose K

Extremal cases

- $K = N$, called leave one out cross validation (loocv)
- $K = 2$

An often-cited problem with loocv is that we have to train many $(= N)$ classifiers, but there is also a deeper problem.

Argument 1: K should be small, e.g. $K = 2$

1. Unless we have a lot of data, variance between two distinct training sets may be considerable.
2. Important concept: By removing substantial parts of the sample in turn and at random, we can simulate this variance.
3. By removing a single point (loocv), we cannot make this variance visible.

How to choose K

Argument 2: K should be large, e.g. $K = N$

1. Classifiers generally perform better when trained on larger data sets.
2. A small K means we substantially reduce the amount of training data used to train each f_k, so we may end up with weaker classifiers.
3. This way, we will systematically overestimate the risk.

How to choose K

Argument 2: K should be large, e.g. $K = N$

1. Classifiers generally perform better when trained on larger data sets.
2. A small K means we substantially reduce the amount of training data used to train each f_k, so we may end up with weaker classifiers.
3. This way, we will systematically overestimate the risk.

Common recommendation: $K = 5$ to $K = 10$

Intuition:

1. $K = 10$ means number of samples removed from training is one order of magnitude below training sample size.
2. This should not weaken the classifier considerably, but should be large enough to make measure variance effects.

How to choose K

Argument 2: K should be large, e.g. $K = N$

1. Classifiers generally perform better when trained on larger data sets.
2. A small K means we substantially reduce the amount of training data used to train each f_k, so we may end up with weaker classifiers.
3. This way, we will systematically overestimate the risk.

Common recommendation: $K = 5$ to $K = 10$

Intuition:

1. $K = 10$ means number of samples removed from training is one order of magnitude below training sample size.
2. This should not weaken the classifier considerably, but should be large enough to make measure variance effects.

How to choose K

Argument 2: K should be large, e.g. $K = N$

1. Classifiers generally perform better when trained on larger data sets.
2. A small K means we substantially reduce the amount of training data used to train each f_k, so we may end up with weaker classifiers.
3. This way, we will systematically overestimate the risk.

Common recommendation: $K = 5$ to $K = 10$

Intuition:

1. $K = 10$ means number of samples removed from training is one order of magnitude below training sample size.
2. This should not weaken the classifier considerably, but should be large enough to make measure variance effects.

How to choose K

Argument 2: K should be large, e.g. $K = N$

1. Classifiers generally perform better when trained on larger data sets.
2. A small K means we substantially reduce the amount of training data used to train each f_k, so we may end up with weaker classifiers.
3. This way, we will systematically overestimate the risk.

Common recommendation: $K = 5$ to $K = 10$

Intuition:

1. $K = 10$ means number of samples removed from training is one order of magnitude below training sample size.
2. This should not weaken the classifier considerably, but should be large enough to make measure variance effects.

How to choose K

Argument 2: K should be large, e.g. $K = N$

1. Classifiers generally perform better when trained on larger data sets.
2. A small K means we substantially reduce the amount of training data used to train each f_k, so we may end up with weaker classifiers.
3. This way, we will systematically overestimate the risk.

Common recommendation: $K = 5$ to $K = 10$

Intuition:

1. $K = 10$ means number of samples removed from training is one order of magnitude below training sample size.
2. This should not weaken the classifier considerably, but should be large enough to make measure variance effects.